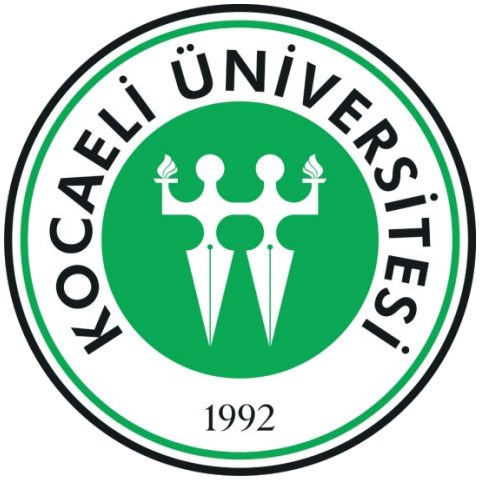


KOCAELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

YAZILIM LABORATUVARI-1
PROJE -3
AKILLI TELEFONLAR İLE NESNE TESPİTİ

ENGİN YENİCE
190201133

KOCAELİ 2021

Akıllı Telefonlar ile Nesne Tespiti

Kocaeli Üniversitesi
Bilgisayar Mühendisliği
Engin Yenice
190201133
enginyenice2626@gmail.com

***Özet*— React Native ve bulut teknolojileri kullanarak seçilen resim üzerinde ki nesnelerin tespit edilmesi amaçlanmaktadır.**

Anahtar Kelimeler—react-native, emulator, nesne tespiti, bulut, compute engine, php, api, android studio, node, npm, npx, google , google cloud, firebase, get, post, fetch

I. GİRİŞ

Projede gerçek zamanlı nesne tespitini React Native kullanarak yapılmıştır Başlangıçta anlık olarak telefon kamerasından çekilen bir görüntüdeki nesneleri tanıyan ve daha sonra görüntü üzerindeki toplam nesne sayısını bulan bir uygulama geliştirilmiştir. Mobil tarafında React Native, API tarafında PHP, bulut olarak Google Cloud kullanılmıştır.

II. KAZANIMLAR

Geliştirilen proje sayesinde

- React native kullanarak mobil program geliştirmeyi
- Bulut teknolojileri ile iletişim kurmayı
- Bulut teknolojileri ile nesne tespiti yapmayı
- Javascript programlama dilinin temel yapısını
- CLI sisteminin kullanımını

öğrendim.

III. EKSİKLER

Talep edilen bütün isterler yapılmıştır. Proje içerisinde herhangi bir eksik bulunma(ma)ktadır.

IV. PROJE YAPISI

Proje 2 katmandan oluşmaktadır. Bunlar:

- React Native Mobil Uygulama
- PHP API

A. React Native Mobil Uygulama

Mobil Uygulama kendi içerisinde 3 sayfadan oluşmaktadır. Bunlar : ana sayfa, tespit edilen resimler ve resim gönder ekranladırı.

1) Ana Sayfa

Giriş sayfasıdır. Uygulamaya giren kullanıcıları bu sayfa karşılamaktadır. Bu ekranda diğer ekranlara geçilebilmesi için butonlar bulunmaktadır. Bu butonlar sayesinde diğer ekranlara geçiş yapabilirsiniz.

2) Resim Gönder

Bu ekran sayesinde kullanıcı tespit etmek istediği nesnenin fotoğrafını, kameradan fotoğraf çekerek veya telefonunda bulunan galeriden resmi seçtiğinde, resim API servisine gönderilir.

Resim API servisine gönderilmesi sırasında internet hızınıza bağlık kısa bir bekleme süresi ile karşılaşırsınız. Bu bekleme süresini belirtmek için sistem tarafından sonuç gelene kadar bunun bilgisini veren bilgi mesajı ile karşılaşırsınız.

İşlem tamamlandığında karşınıza tespit edilmesi için gönderdiğiniz resmin üzerinde tespit edilen nesneler dikdörtgen ile çerçeve içerisine alınır ve size gösterilir. Gösterilen resim üzerinde yakınlaştırma ve uzaklaştırma (Ekran üzerine 2 parmağınızı koyup zıt kenarlara uzaklaştırmanız sonucunda yakınlaştırma. Ekran üzerine 2 parmağınızı koyup ekranın ortasına doğru yakınlaştırmanız sonucunda uzaklaştırma) işlemi yapabilirsiniz.

Resim hakkında daha detaylı bilgiye ulaşmak için resim detayları butonuna tıklayabilirsiniz. Yeni açılan ekran içerisinde resim içerisinde tespit edilen resimlerin isimleri ve toplam nesne sayısını göstermektedir.

3) Tespit Edilen Resimler

Bu ekranda nesne tespiti yapılmış ve firebase veri tabanına kayıt edilmiş tüm resimlerin kayıtlarına ulaşabileceğiniz temel bir galeri mantığı ile oluşturulmuştur.

Bu ekran üzerinde incelemek istediğiniz resme dokunarak resmi tam ekran olarak görebilirsiniz. Tam ekran olarak açılan resmi sağdan sola doğru kaydırarak nesne tespitinin yapılmamış haline ulaşabilirsiniz. Resim üzerinde yakınlaştırma ve uzaklaştırma (Ekran üzerine 2 parmağınızı koyup zıt kenarlara uzaklaştırmanız sonucunda yakınlaştırma. Ekran üzerine 2 parmağınızı koyup ekranın ortasına doğru yakınlaştırmanız sonucunda uzaklaştırma) işlemi yapabilirsiniz.

Resim hakkında daha detaylı bilgiye ulaşmak için resim detayları butonuna tıklayabilirsiniz. Yeni açılan ekran içerisinde resim içerisinde tespit edilen resimlerin isimleri ve toplam nesne sayısını göstermektedir.

4) Kullanılan Paketler

React Native CLI paketlerinin dışında 2 adet ayrı paket kullanılmıştır.

a) React-native-image-picker

Kamera üzerinden ve galeri üzerinden resim seçilmesi için kullanılmış bir pakettir.

b) React-native-image-zoom-viewer

Resimler üzerinde yakınlaştırma, uzaklaştırma ve yan yana birden çok resim koyma gibi özelliklerinden kullanılmak amacıyla kullanılmıştır.

B. PHP API

Mobil uygulama tarafından POST metodunu yakalar. Gönderilen POST metodu içerisinde bulunan resmi ”normalImages” klasörü içerisine kayıt eder. Kayıt edilen resim, nesne tespiti yapılması için Application adında oluşturduğum sınıfın nesne tespiti fonksiyonuna gönderilir. Bu fonksiyon 3 adet parametre almaktadır (Resim URL, Resim Yolu, Resim Tipi).

Nesne tespiti fonksiyonu çalıştığında **Google Vision API** ile haberleşerek resim üzerinde bulunan nesnelerin tespitini yapar. Buradan gelen JSON tipinde bilgiler ile resim üzerinde dikdörtgen çizimi yapılır.

NOT: Koordinat bilgileri 0,1 arasında bir değer gelmektedir. Bu değerleri px cinsine çevirmek için

X koordinatı için = Resin genişliği * (0-1 arasında gelen değer)

Y koordinatı için = Resim yüksekliği * (0-1 arasında gelen değer)

Resim çizim işlemi gerçekleştikten sonra kullanıcının gönderdiği resim, yeni oluşturulan resim firebase storage kayıt edilir. Resimler firebase storage kayıt edilmesinin ardından sunucu üzerinden silinir.

Kullanıcının gönderdiği resimin firebase storage linki, tespit edilen nesnelerin dikdörtgen içerisine aldındığı firebase storage linki ve tespit edilen resimlerin isimlerinin bulunduğu bir json objesi oluşturulur ve bu obje API aracılığı ile react native tarafına gönderilir.

C. Kullanılan Paketler

1) google/cloud-vision
Google Vision API ile haberleşmemizi sağlayan paket

2) kreait/firebase-php
Firebase işlemlerinin gerçekleştirilmesinde kullanılan paket

V. KLASÖR YAPISI VE DOSYA ÖZELLİKLERİ

A. PHP-API

```
PHP-API
├── detectedImages/
├── normalImages/
├── secret/
│   ├── key.json
│   └── firebase-key.json
├── vendor/
├── composer.json
├── index.php
├── db.php
└── storage.php
```

1) detectedImages
Tespit edilen resmin otomatik olarak yükleneceği klasör. Kesinlikle bulunmalıdır.

2) normalImages
Kullanıcının gönderdiği resmin tutulduğu klasör. Kesinlikle bulunmalıdır.

3) secret
Google Cloud Vision API ve Firebase SDK API keylerinin bulunduğu klasördür.

4) vendor
Projeye ait paketlerin bulunduğu klasör.

5) composer.json
Projenin bağımlılıklarının liste olarak tutulduğu json dosyasıdır.

6) index.php
Projenin temel haberleşme işlemleri buradan yapılmaktadır. Bu sayfa içerisine dosya tipinde bir post işlemi gerçekleştiğinde, gelen resmi benzersiz bir isim vererek normalImages klasörüne kayıt eder. Kayıt edilen resmi nesne tespiti için application.php dosyasına gönderir. Tespit edilen resimler hakkında geri dönüş aldıktan sonra bu bilgileri db.php dosyasına göndererek firebase kayıt işlemlerini yaptırır. Son olarak tüm bilgileri birleştirerek gelen POST işlemine json formatında geri dönüş yapar.

7) db.php
Kendisine gönderilen JSON formatındaki datayı düzenler ve firebase realtimer database sistemine kayıt eder. Arından resimler storage yüklenmesi için storage.php klasörüne gönderilir. Resim yükleme işlemi tamamlandıktan sonra detectedImages ve normalImages klasörleri içerisine kayıt edilen resimler silinir.

8) application.php
İlk olarak VisionClient tanımlanır. Bu sınıf tanımlanırken Google Cloud Vision API tarafından verilen JSON dosyasının yolu gösterilir. Yapılan bu işlemle api ile iletişim kurulması sağlanır. Ardından tespit edilecek resim api gönderilir. Tespit işlemi tamamlandığında API dan gelen JSON formatındaki veriye göre resim üzerinde çizim işlemleri gerçekleşir ve resim detectedImages klasörü içerisine kayıt edilir. Son olarak resmin yolu ve nesnelerin bilgileri index.php ye JSON formatında gönderilir.

9) storage.php
İlk olarak Factory tanımlanır. Bu sınıf tanımlanırken Firebase API tarafından verilen JSON dosyasının yolu gösterilir. Yapılan bu işlemle api ile iletişim kurulması sağlanır. Gönderilen resimler Firebase Storage yüklenir. Yüklenen resimler ile ilgili url bilgisi geri döndürülür.

B. ReactNative-APP

```
ReactNative-APP
├── Screen/
│   ├── Gallery.js
│   ├── Home.js
│   └── ObjectDetection.js
├── package.json
└── App.js
```

1) Screen/Gallery.js
Firebase üzerine fetch ile gidilir ve orada bulunan tüm kayıtları alır. Alınan kayıtları galeri formatında ekranda gösterir.

2) Screen/Home.js

Projenin ana ekranını göstermektedir. Bu ekran üzerinden nesne tespitine işlemine ya da proje galerisine gidebileceğiniz butonlar bulunur.

3) Screen/ObjectDetection.js

Bu ekran üzerinden cihazın galerisinden seçilen resmi ya da kamera ile anlık olarak çekilen resmi API ye gönderilir. Gelen bilgi ile ekrana resim ile ilgili bilgiler çizilir.

4) Package.json

Proje bağımlılıklarının ve komutlarının listesinin bulunduğu json dosyasıdır.

5) App.js

Ekranda hangi ekranın gösterileceğini belirtir. İlk olarak Home.js ekranı gösterilir.

VI. Proje Bağımlılıkları

A. PHP

1) *Composer*
2) *PHP 7.3 ve üzerinde*

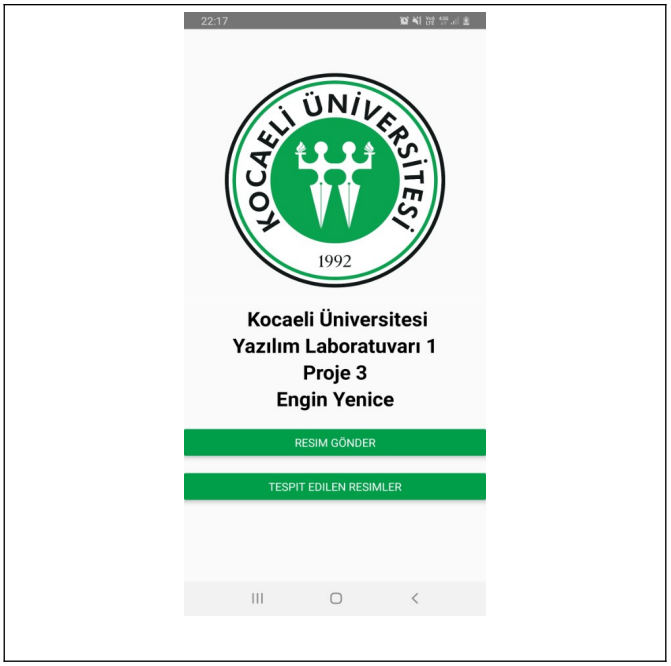
B. React Native

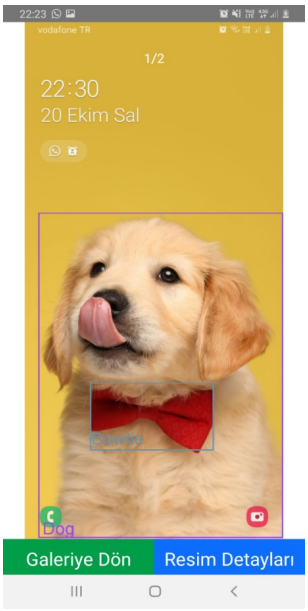
1) *Node*
2) *NPM*
3) *React Native CLI*

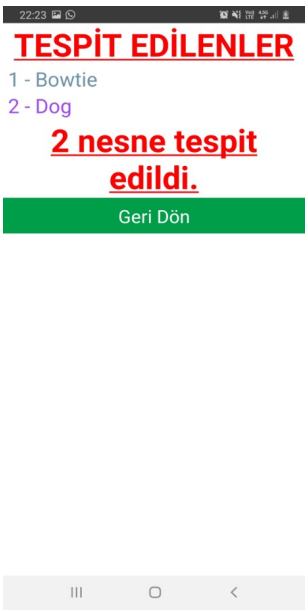
C. Android Studio

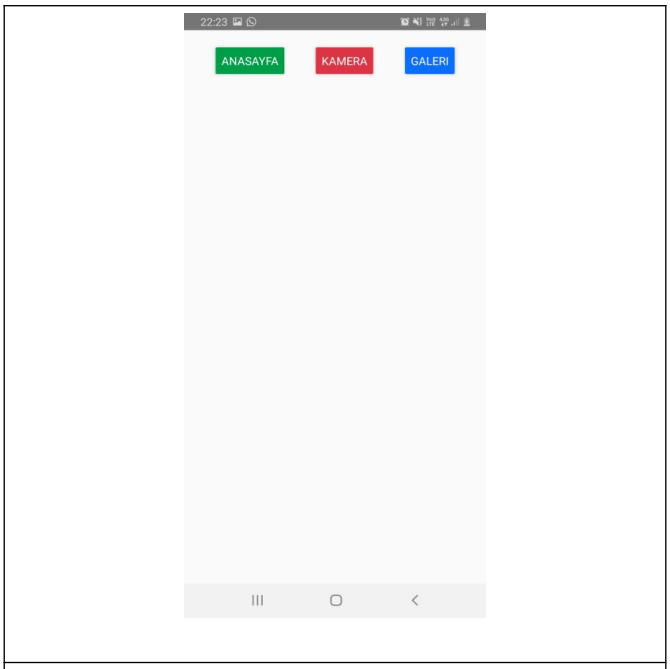
1) *Android 10 (Q)*
2) *Android SDK Platform 29*
3) *Google APIs Intel x86 Atom_64 System Image*
4) *Intel x86 Atom_64 System Image*
5) *Android SDK Build- Tools 29.0.3*

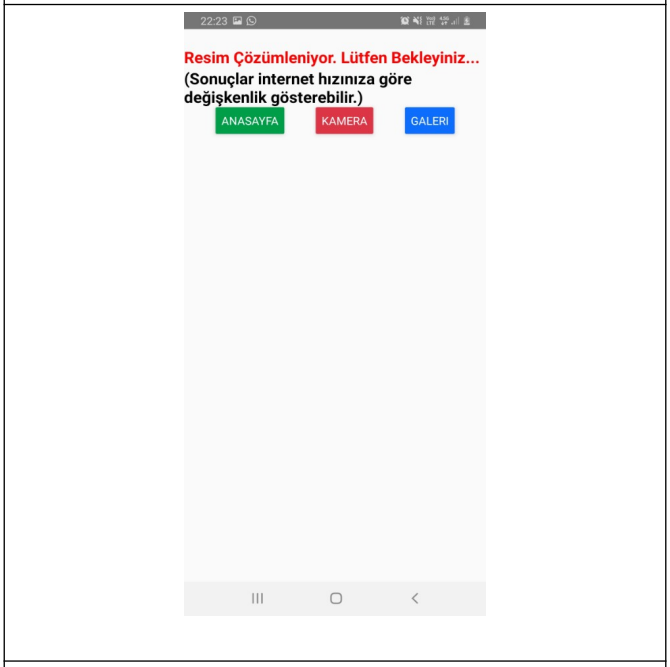
VII. Resimler

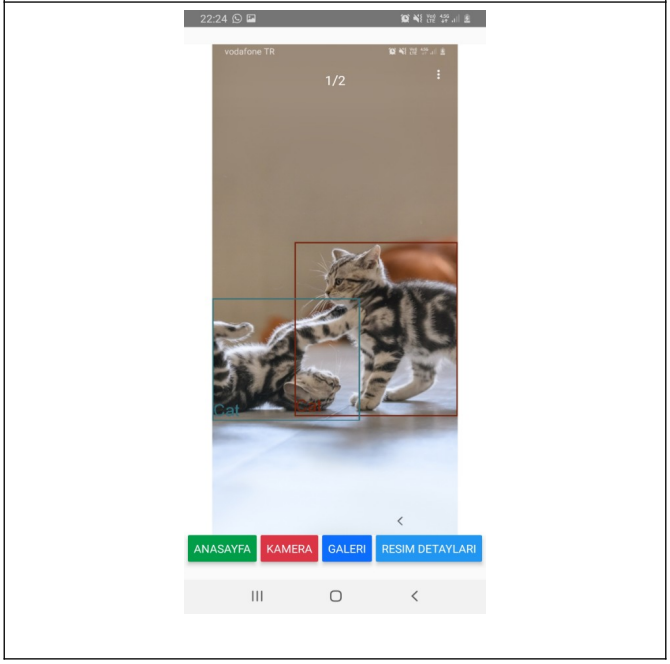












VIII. KABA KOD

190201133-Kaba-Kod.pdf dosyası içerisinde yer almaktadır.

IX. KAYNAKÇA

1. https://www.cyberciti.biz/faq/star-stop-restart-apache2-webserver/
2. https://github.com/ivpusic/react-native-image-crop-picker/issues/1098
3. https://www.youtube.com/watch?v=Uae7mVQJ_4c&list=PLC-R40l2hJfeaLSr8C-QV3o6xCtIJm7uL&index=5
4. https://cloud.google.com/vision/docs/object-localizer#vision_localize_objects_gcs-nodejs
5. https://reactnative.dev/
6. http://enginyenice.com/
7. https://firebase.google.com/
8. https://cloud.google.com/
9. https://stackoverflow.com/questions/6602422/access-denied-when-creating-keystore-for-android-app
10. http://mobilkodakademi.com/react-native/react-native-apk-olusturma-release-signing-apk/
11. https://www.cyberciti.biz/faq/star-stop-restart-apache2-webserver/